UNCLASSIFIED

TARDEC

---TECHNICAL REPORT---

No. **21187**

By: J. Goryca

Force and Moment Plots from Pacejka 2002 Magic Formula Tire Model Coefficients

Distribution: **STATEMENT A. Approved for public release; distribution is unlimited.**

U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research Development and Engineering Center (TARDEC)
Detroit Arsenal
6501 East 11 Mile Road
Warren, Michigan 48397-5000

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
10 SEP 2010	**N/A**	**-**

4. TITLE AND SUBTITLE
Force and Moment Plots from Pacejka 2002 Magic Formula Tire Model Coefficients

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Jill E. Goryca

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Tank-automotive and Armaments Research, Development and Engineering Center ATTN: RDTA-RS/MS157 6501 E 11 Mile Road Warren, MI 48397-5000

8. PERFORMING ORGANIZATION REPORT NUMBER
21187RC

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Tank-automotive and Armaments Research, Development and Engineering Center ATTN: RDTA-RS/MS157 6501 E 11 Mile Road Warren, MI 48397-5000

10. SPONSOR/MONITOR'S ACRONYM(S)
TACOM/TARDEC

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
21187RC

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT
One of the important aspects in vehicle dynamics simulation is accurate modeling of the tire-road interaction forces because the movement of the vehicle depends on the forces and moments applied to the tires. Many vehicle simulation programs such as MSC.Software ADAMS, Altair HyperWorks MotionSolve, etc. use the Magic Formula (MF) developed by Pacejka to model tires. In order to use these applications more effectively, a tool was developed using MATLAB to quickly calculate and plot the forces and moments represented by the tire coefficients. The tool is consistent with expected results in determining the tire-road interaction forces. This report describes the structure of the program and how to use the Pacejka Plot Tool to plot the forces and moments from the coefficients using the PAC2002 MF tire model.

15. SUBJECT TERMS
Magic Formula; PAC2002; MATLAB GUIDE; longitudinal force; lateral force; overturning moment; rolling resistance moment; self-aligning moment

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**SAR**	**39**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*
September 2010

2. REPORT TYPE
Technical

3. DATES COVERED *(From - To)*
July 2010 - September 2010

4. TITLE AND SUBTITLE
Force and Moment Plots from Pacejka 2002 Magic Formula Tire Model Coefficients

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Goryca, Jill E.

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
U.S. Army Tank-automotive and Armaments Research, Development and Engineering Center
ATTN: RDTA-RS/MS157
6501 E 11 Mile Road
Warren, MI 48397-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
U.S. Army Tank-automotive and Armaments Research, Development and Engineering Center
6501 E 11 Mile Road
Warren, MI 48397-5000

10. SPONSOR/MONITOR'S ACRONYM(S)
RDECOM-TARDEC

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution Statement A. Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

One of the important aspects in vehicle dynamics simulation is accurate modeling of the tire-road interaction forces because the movement of the vehicle depends on the forces and moments applied to the tires. Many vehicle simulation programs such as MSC.Software ADAMS, Altair HyperWorks MotionSolve, etc. use the Magic Formula (MF) developed by Pacejka to model tires. In order to use these applications more effectively, a tool was developed using MATLAB to quickly calculate and plot the forces and moments represented by the tire coefficients. The tool is consistent with expected results in determining the tire-road interaction forces. This report describes the structure of the program and how to use the Pacejka Plot Tool to plot the forces and moments from the coefficients using the PAC2002 MF tire model.

15. SUBJECT TERMS
Magic Formula; PAC2002; MATLAB GUIDE; longitudinal force; lateral force; overturning moment; rolling resistance moment; self-aligning moment

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
					Goryca, Jill E.
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	Unclassified Unlimited	39	**19b. TELEPHONE NUMBER** *(include area code)*
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED			

1.0 INTRODUCTION 1
2.0 BACKGROUND 1
3.0 STRUCTURE OF MATLAB PROGRAM 2
4.0 HOW TO USE THE GUI 4
5.0 SUMMARY/CONCLUSION 5
6.0 CONTACT 5
7.0 REFERENCES 5
8.0 DEFINITIONS, ACRONYMS, ABBREVIATIONS 6
9.0 DISTRIBUTION 6
APPENDIX A: SAMPLE TIRE DATA INPUT FILE A-1
APPENDIX B: MAGIC FORMULA EQUATIONS [1,3] B-1
APPENDIX C: MATLAB FUNCTION FILES C-1

FIGURE 1 – DIAGRAM OF FORCES, MOMENTS AND ANGLES IN TIRE MODEL [4] 1
FIGURE 2 – CHARACTERISTIC CURVES FOR FX AND FY UNDER PURE SLIP CONDITIONS [1] 2
FIGURE 3 – INTERACTION OF MATLAB FILES 3
FIGURE 4 – FEATURES OF THE PACEJKA PLOT TOOL 4

TABLE 1 – DESCRIPTION OF FUNCTION M-FILES 2
TABLE 2 – DESCRIPTION OF INPUT VARIABLES 3
TABLE 3 – PLOT TYPE OPTIONS 4

Technical Report 21187
September 2010

Force and Moment Plots from Pacejka 2002 Magic Formula Tire Model Coefficients

Jill Goryca
U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research, Development and Engineering Center (TARDEC)
Concepts, Analytics, System Simulation & Integration (CASSI) Dynamics and Structures Team
ATTN: RDTA-RS/MS157
6501 E 11 Mile Road
Warren, Michigan 48397-5000

1.0 INTRODUCTION

One of the important aspects in vehicle dynamics simulation is accurate modeling of the tire-road interaction forces because the movement of the vehicle depends on the forces and moments applied to the tires [1]. Many vehicle simulation programs such as MSC.Software ADAMS, Altair HyperWorks MotionSolve, etc. use the Magic Formula (MF) developed by Pacejka to model tires. In order to use these applications more effectively, a tool was developed using MATLAB to quickly calculate and plot the forces and moments represented by the tire coefficients. The tool is consistent with expected results in determining the tire-road interaction forces. This report describes the structure of the program and how to use the Pacejka Plot Tool to plot the forces and moments from the coefficients using the PAC2002 MF tire model.

2.0 BACKGROUND

A diagram of the forces and moments calculated from the MF is shown in Figure 1. The longitudinal force, lateral force, overturning moment, rolling resistance moment, and self-aligning moment are calculated from the vertical force Fz, inclination angle γ, slip angle α, and longitudinal slip κ. The longitudinal slip, which is not shown in the diagram, depends on the longitudinal velocity of the axle, the effective radius of the tire, and the angular velocity of the tire.

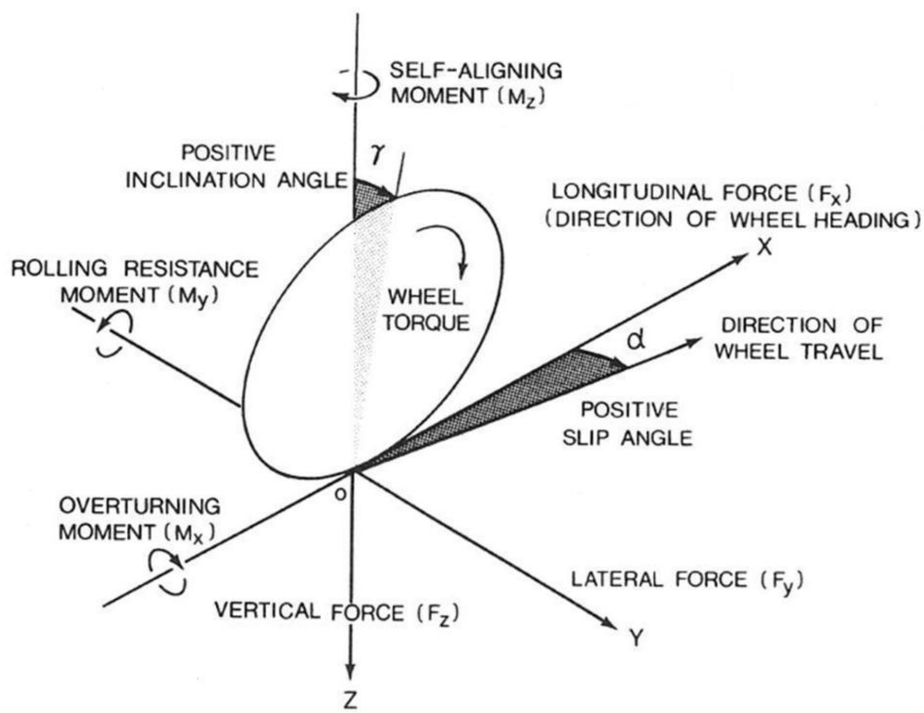

Figure 1 – Diagram of forces, moments and angles in tire model [4]

The forces and moments were plotted using MATLAB software. This versatile software can be used for many applications including data processing, data analysis and data visualization. The Graphical User Interface Development Environment (GUIDE) was used to develop the Pacejka Plot Tool, because a GUI enables the user to easily change features of the plot without having to write MATLAB commands. In order to run the Pacejka Plot Tool, MATLAB must be installed on the user's computer.

The tire coefficients used in the Magic Formula are determined from a curve that is fitted to experimental data. The tire coefficients that are generated from this curve are stored in a tire data file. The tire data files are designated with the extension of *.tir. The MATLAB program searches for files with the *.tir extension and displays them in a list box. When the coefficients are needed in the Magic Formula, the values of the coefficients are parsed from the selected tire data file. The basic format of the tire data file is COEF = 5.000e-002. This is shown in the sample input file included in Appendix A. Due to the proprietary nature of this type of tire data, fictitious tire data was used in the sample.

The general equation for the magic formula is taken from [1] and is shown in equation (1) where F(x) is either Fx with x the longitudinal slip κ, or Fy, and x the lateral slip α. The coefficients B, C, D, and E are calculated from additional equations and the coefficients found in the tire data files. The complete set of equations that was used is included in Appendix B.

$$F(x) = D \cos\left[C \arctan\{Bx - E(Bx - \arctan(Bx))\}\right] \qquad (1)$$

The characteristic curves for the longitudinal force, Fx, and the lateral force, Fy, are shown in Figure 2. These characteristic curves were used to debug the code and verify that the correct approach was used.

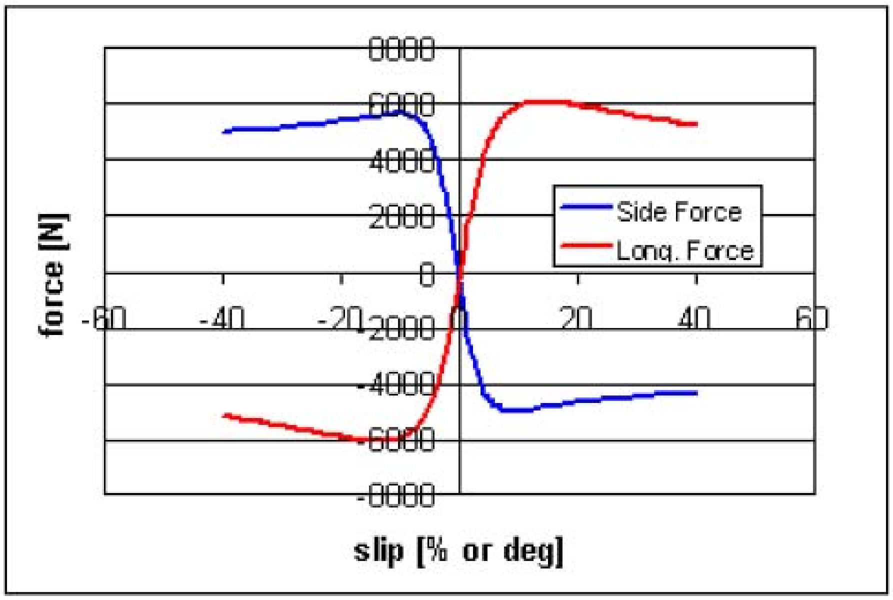

Figure 2 – Characteristic curves for Fx and Fy under pure slip conditions [1]

The Pacejka Plot Tool can calculate the longitudinal force and the lateral force using pure slip or combined slip. This option is determined by the value of the Use Mode in the tire data file. If the Use Mode is equal to 3, the forces are calculated using pure slip. If the Use Mode option is equal to 4, the forces are calculated for combined slip. One of the differences between combined slip and pure slip that combined slip will cause the longitudinal and lateral forces to decrease compared to pure slip. In addition, more tire coefficients are required to calculate the forces using combined slip.

3.0 STRUCTURE OF MATLAB PROGRAM

Each function in MATLAB has a separate file with a .m extension. The name of the m-file is the same as the name of the function. Since the desired result of the program was to plot the forces and moments as a function of the user's input, it was deemed appropriate to calculate the forces and moments using function m-files. Therefore, in order to run the tool, it is necessary to have the supporting function files in the same folder as the Pacejka Plot Tool file. A description of each file is given in Table 1. For future reference, a description of the input variables used by the functions is given in Table 2. The interaction between the functions is shown in Figure 3.

Table 1 – Description of function m-files

Function/File Name	Description
Pacejka_Plot_Tool	Main program that calls functions based on user action. Plots forces and moments.
ImportTireData	Reads tire coefficients from file into MATLAB array.
gvar	Gets specific tire coefficients from array.

Fx	Calculates the longitudinal force.
Fy	Calculates the lateral force.
F	Calculates magic formula. This function is used by both Fx and Fy.
fG	Calculates combined slip factor when Use Mode is equal to 4.
MomentCalc	Calculates overturning moment, rolling resistance moment, and self-aligning moment.

Table 2 – Description of input variables

Input Variables	Description
varargin	Used to input a variable number of arguments. The purpose of varargin is explained in the comments of each function.
filename	The name and path of the input tire data file.
varname	The text string of the tire coefficient name.
array	A cell array containing the tire coefficient name and value.
kappa	Longitudinal slip
Fz	Vertical force
gamma	Inclination angle
alpha	Slip angle
isX	Flag to distinguish calculation of Fx or Fy

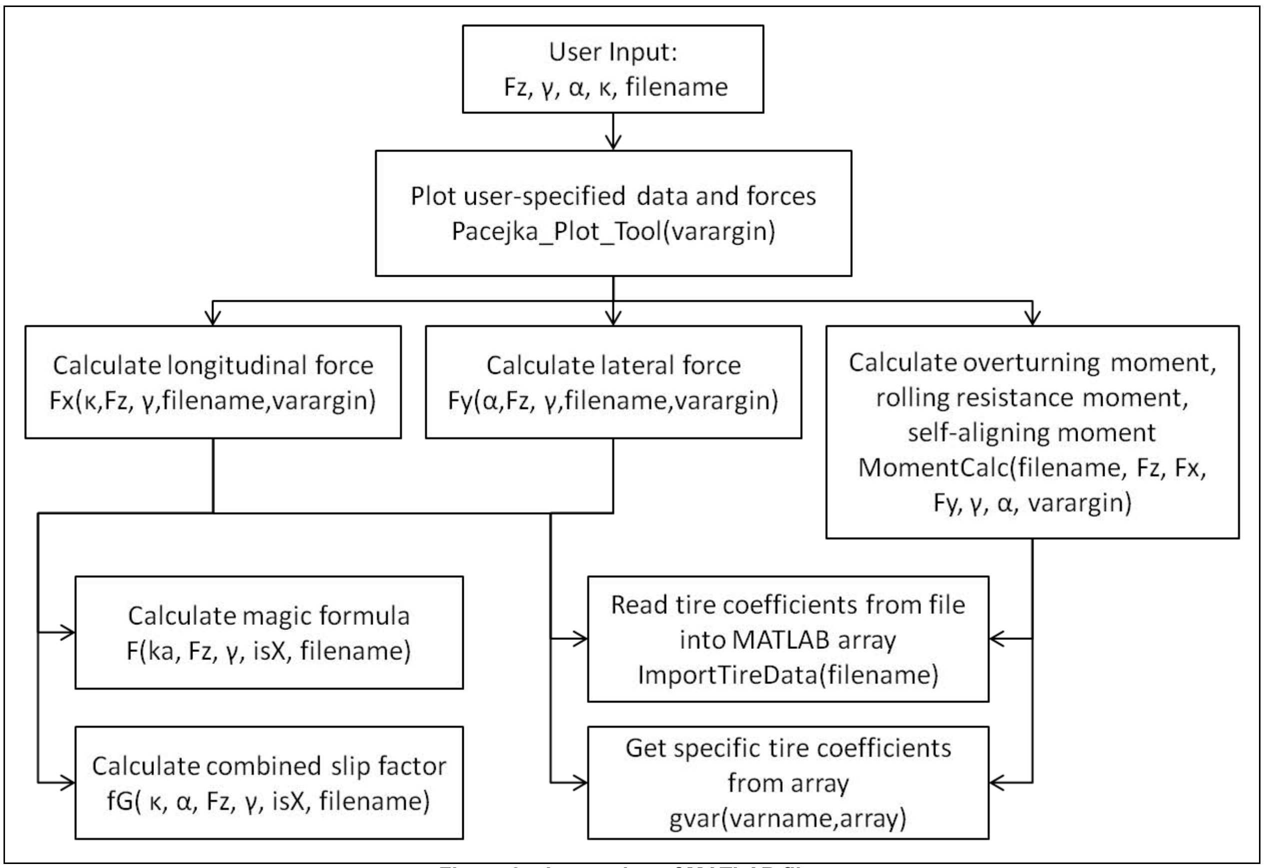

Figure 3 – Interaction of MATLAB files

4.0 HOW TO USE THE GUI

The Pacejka Plot Tool uses SI units for all calculations and text inputs. English units are provided in some cases for the user's convenience.

To open the tool, first open MATLAB. Change the current folder in MATLAB to the location of the MATLAB files. Right-click on "Pacejka_Plot_Tool.m" and choose "Run" to open the program. If the tire data files are not in the current folder or the folder titled "Tire Data", the user will be asked to select the location and an open file dialog will appear. Figure 4 shows the Pacejka Plot Tool GUI with each feature numbered for reference. A description of the features follows the figure.

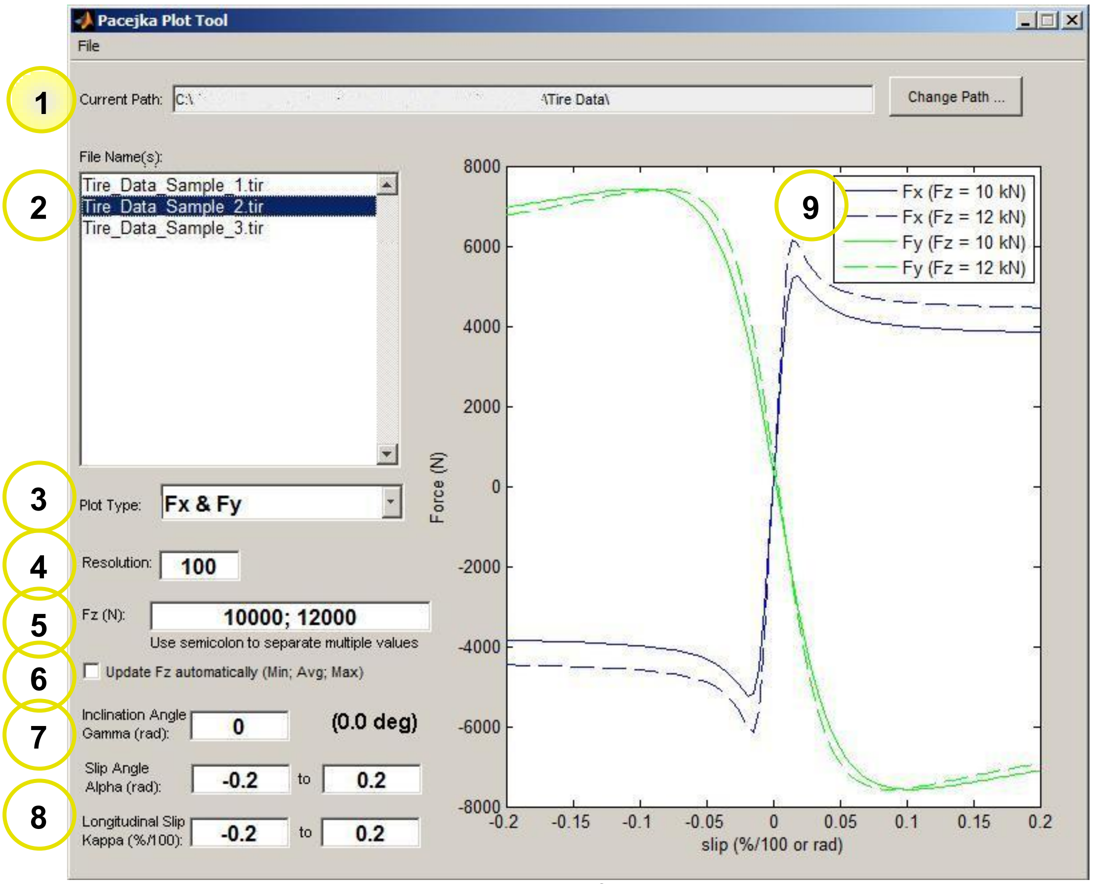

Figure 4 – Features of the Pacejka Plot Tool

1) To use a tire data file that is not in the current folder, click on the "Change Path…" button. This will reopen the file dialog box to browse for a tire data file (*.tir). After a tire data file is selected, any other tire data files in that folder will also be displayed in the list box.

2) To plot a file shown in the list box, simply select the file name. Multiple selections can be made by holding the SHIFT or CTRL key.

3) To select the forces and moments to be plotted, change the plot type. A description of the available options is shown in Table 3. Each plot type plots all of the user-specified values of Fz using a different color for each type of force and moment. The colors remain consistent when changing plot types. For example, selecting the option Fx & Fy plots Fx in blue and Fy in green (as shown in Figure 4). If the option Fy is selected, the blue lines on the graph in Figure 4 would be removed, but the plot of Fy would remain green.

Table 3 – Plot Type Options

Plot Option	Description
Fx & Fy	Plots the longitudinal force and the lateral force
Mx, My, & Mz	Plots the overturning moment, rolling resistance moment, and the self-aligning moment
Fx	Plots the longitudinal force only

Fy	Plots the lateral force only
Mx	Plots the overturning moment only
My	Plots the rolling resistance moment only
Mz	Plots the self-aligning moment only
All	Plots the longitudinal force, lateral force, overturning moment, rolling resistance moment, and the self-aligning moment

4) The resolution is the number of steps between the upper and lower limits of the graph. If the plot does not look like a smooth curve, increase the resolution until the plot looks smoother. This increases the number of points plotted in the graph.
5) To specify the vertical force to input into the magic formula, type a number in the Fz text box. To input multiple values of Fz, enter each value separated by a semicolon. The forces and moments specified by the plot type will be plotted for each vertical force input. Adding another vertical force input adds a line of the same color as the original line, with a different line type to distinguish it from the preceding graph. For example, if the vertical force, Fz, was equal to 10,000 N, the plot in Figure 3 would show just the solid lines. Since there are two vertical force inputs in Figure 3, 10,000 N and 12,000 N, two lines of each color are shown. The legend indicates the longitudinal force, Fx, or lateral force, Fy, as well as the value for Fz (kN) for each line. The line types cycle through solid, dashed, dotted, and dash-dot. If more than four forces are input, the line type cycles back to solid.
6) The Update Fz Automatically check box changes the vertical force input based on the values of FzMin and FzMax in the selected tire data file. Checking this box will overwrite the current value of the Fz text box with three values: FzMin, the average of FzMin and FzMax, and FzMax. When two or more files are selected, the values from the first file selected are used to automatically update the force.
7) To vary the inclination angle, γ, enter a value in the text box. Although the inclination angle is entered in radians, it is converted to degrees and shown in parentheses next to the text box for the user's convenience.
8) To change the limits of the horizontal axis, adjust the ranges for slip angle, α, and longitudinal slip, κ. The longitudinal force is plotted between the lower and upper limits of the longitudinal slip and the lateral force is plotted between the lower and upper limits of the slip angle (rad). The longitudinal slip is a ratio with units of %/100 (i.e. 20% = 0.2).
9) If the legend obscures the graph, it can be easily moved by clicking and dragging with the mouse.

5.0 SUMMARY/CONCLUSION

The Pacejka Plot Tool plots the longitudinal force, lateral force, overturning moment, rolling resistance moment, and self-aligning moment from PAC2002 MF using the tire coefficients data file and user input. As discussed in Section 4.0, the user can specify the file name and location, plot type, vertical force, inclination angle, slip angle, and longitudinal slip to customize the plot. Although the intended purpose of this tool is to plot tire data, with additional modifications, the GUI could be reused to plot other types of data read from a file.

6.0 CONTACT

The author is a co-op engineer with the U.S. Army Research, Development and Engineering Command (RDECOM), located at the U.S. Army Tank-automotive and Armaments Research, Development and Engineering Center (TARDEC). Interested parties can contact the author at the title page address or email: jill.goryca@us.army.mil.

7.0 REFERENCES

[1] MSC.Software PAC2002 ADAMS/Tire 2005 r2 Help pp. 22-96
http://ti.mb.fh-osnabrueck.de/adamshelp/mergedProjects/tire/Chapter_04_PAC2002.pdf

[2] I.J.M. Besselink (TU/e), A.J.C. Schmeitz (TNO) and H.B. Pacejka (TU Delft) "An Improved Magic Formula/Swift Tyre Model that can Handle Inflation Pressure Changes" Department of Mechanical Engineering, Eindhoven University of Technology P.O. Box 513, 5600 MB Eindhoven, the Netherlands, e-mail address of lead author: i.j.m.besselink@tue.nl
http://www.mate.tue.nl/mate/showabstract.php/11281

[3] MathWorks, Inc. MATLAB R2009a help file

[4] Wong, J.Y., Theory of Ground Vehicles 3rd ed., John Wiley & Sons, 2001. ISBN 0-471-35461-9.

8.0 DEFINITIONS, ACRONYMS, ABBREVIATIONS

RDECOM – U.S. Army Research, Development and Engineering Command

TACOM – U.S. Army Tank-automotive and Armaments Command

TARDEC – U.S. Army Tank-automotive and Armaments Research, Development and Engineering Center

GUI – Graphical User Interface

MATLAB – Matrix Laboratory

SI – Système international d'unités (International System of Units)

MF – Magic Formula

PAC2002 – Pacejka 2002

9.0 DISTRIBUTION

Statement A: Approved for public release; distribution is unlimited.

APPENDIX A: SAMPLE TIRE DATA INPUT FILE

```
$---------------------------------------------
INFLATION_PRESSURE        =        5
$---------------------------------------------units
[UNITS]
LENGTH                =        'meter'
FORCE                 =        'newton'
ANGLE                 =        'radians'
MASS                  =        'kg'
TIME                  =        'second'
$---------------------------------------------model
! USE_MODE specifies the type of calculation performed:
!      3: Fx,Fy,Mx,My,Mz uncombined force/moment calculation
!      4: Fx,Fy,Mx,My,Mz combined force/moment calculation
!---------------------------------------------
[MODEL]
USE_MODE              =        3        $
FITTYP                =        5        $
VXLOW                 =        1
LONGVL                =        1        $Measurement speed
$---------------------------------------------dimensions
[DIMENSION]
UNLOADED_RADIUS       =        0.5000  $Free tyre radius
WIDTH                 =        0.5000  $Nominal section width of the tyre
ASPECT_RATIO          =        0.5000  $Nominal aspect ratio
RIM_RADIUS            =        0.5000  $Nominal rim radius
RIM_WIDTH             =        0.5000  $Rim width
$---------------------------------------------parameter
[VERTICAL]
VERTICAL_STIFFNESS =        5.0000e+005          $Tyre vertical stiffness
VERTICAL_DAMPING   =        50                   $Tyre vertical damping
BREFF                 =        5.000                $Low load stiffness e.r.r.
DREFF                 =        0.500                $Peak value of e.r.r.
FREFF                 =        0.050                $High load stiffness e.r.r.
FNOMIN                =        15000                $Nominal wheel load
$---------------------------------------------long_slip_range
[LONG_SLIP_RANGE]
KPUMIN                =        -0.50000             $Minimum valid wheel slip
KPUMAX                =        0.50000              $Maximum valid wheel slip
$---------------------------------------------slip_angle_range
[SLIP_ANGLE_RANGE]
ALPMIN                =        -0.50000             $Minimum valid slip angle
ALPMAX                =        0.50000              $Maximum valid slip angle
$---------------------------------------------inclination_slip_range
[INCLINATION_ANGLE_RANGE]
CAMMIN                =        -0.50000             $Minimum valid camber angle
CAMMAX                =        0.50000              $Maximum valid camber angle
$---------------------------------------------vertical_force_range
[VERTICAL_FORCE_RANGE]
FZMIN                 =        5000     $Minimum allowed wheel load
FZMAX                 =        20000    $Maximum allowed wheel load
$---------------------------------------------scaling
[SCALING_COEFFICIENTS]
LFZO                  =        1        $Scale factor of nominal (rated) load
LCX                   =        1        $Scale factor of Fx shape factor
LMUX                  =        1        $Scale factor of Fx peak friction coefficient
LEX                   =        1        $Scale factor of Fx curvature factor
LKX                   =        1        $Scale factor of Fx slip stiffness
LHX                   =        1        $Scale factor of Fx horizontal shift
```

LVX	=	1	$Scale factor of Fx vertical shift
LCY	=	1	$Scale factor of Fy shape factor
LMUY	=	1	$Scale factor of Fy peak friction coefficient
LEY	=	1	$Scale factor of Fy curvature factor
LKY	=	1	$Scale factor of Fy cornering stiffness
LHY	=	1	$Scale factor of Fy horizontal shift
LVY	=	1	$Scale factor of Fy vertical shift
LGAY	=	1	$Scale factor of camber for Fy
LTR	=	1	$Scale factor of Peak of pneumatic trail
LRES	=	1	$Scale factor for offset of residual torque
LGAZ	=	1	$Scale factor of camber for Mz
LXAL	=	1	$Scale factor of alpha influence on Fx
LYKA	=	1	$Scale factor of alpha influence on Fx
LVYKA	=	1	$Scale factor of kappa induced Fy
LS	=	1	$Scale factor of Moment arm of FxL
LSGKP	=	1	$Scale factor of Relaxation length of Fx
LSGAL	=	1	$Scale factor of Relaxation length of Fy
LGYR	=	1	$Scale factor of gyroscopic torque
LMX	=	1	$Scale factor of overturning couple
LMY	=	1	$Scale factor of rolling resistance torque

$--longitudinal
[LONGITUDINAL_COEFFICIENTS]

PCX1	=	1.5000e+000	$Shape factor Cfx for longitudinal force
PDX1	=	5.0000e-001	$Longitudinal friction Mux at Fznom
PDX2	=	-5.0000e-002	$Variation of friction Mux with load
PEX1	=	-5.0000e+000	$Longitudinal curvature Efx at Fznom
PEX2	=	-5.0000e+000	$Variation of curvature Efx with load
PEX3	=	5.0000e-002	$Variation of curvature Efx with load squared
PEX4	=	0.0000e+000	$Factor in curvature Efx while driving
PKX1	=	5.0000e+001	$Longitudinal slip stiffness Kfx/Fz at Fznom
PKX2	=	5.0000e-002	$Variation of slip stiffness Kfx/Fz with load
PKX3	=	-5.0000e-002	$Exponent in slip stiffness Kfx/Fz with load
PHX1	=	0.0000e+000	$Horizontal shift Shx at Fznom
PHX2	=	0.0000e+000	$Variation of shift Shx with load
PVX1	=	-0.0000e+000	$Vertical shift Svx/Fz at Fznom
PVX2	=	0.0000e+000	$Variation of shift Svx/Fz with load
RBX1	=	5.0000e+000	$Slope factor for combined slip Fx reduction
RBX2	=	5.0000e+000	$Variation of slope Fx reduction with kappa
RCX1	=	5.0000e+000	$Shape factor for combined slip Fx reduction
RHX1	=	0.0000e+000	$Shift factor for combined slip Fx reduction
PTX1	=	0.0000e+000	$Relaxation length SigKap0/Fz at Fznom
PTX2	=	0.0000e+000	$Variation of SigKap0/Fz with load
PTX3	=	0.0000e+000	$Variation of SigKap0/Fz with exponent of load

$--overturning
[OVERTURNING_COEFFICIENTS]

QSX1	=	0.0000e+000	$Lateral force induced overturning moment
QSX2	=	0.0000e+000	$Camber induced overturning couple
QSX3	=	0.0000e+000	$Fy induced overturning couple

$--lateral
[LATERAL_COEFFICIENTS]

PCY1	=	1.5000e+000	$Shape factor Cfy for lateral forces
PDY1	=	5.0000e-001	$Lateral friction Muy
PDY2	=	-5.0000e-001	$Variation of friction Muy with load
PDY3	=	-5.0000e+000	$Variation of friction Muy with squared camber
PEY1	=	5.0000e-002	$Lateral curvature Efy at Fznom
PEY2	=	-5.0000e-003	$Variation of curvature Efy with load
PEY3	=	5.0000e-001	$Zero order camber dependency of curvature Efy
PEY4	=	5.0000e+002	$Variation of curvature Efy with camber
PKY1	=	-5.0000e+001	$Maximum value of stiffness Kfy/Fznom
PKY2	=	5.0000e+000	$Load at which Kfy reaches maximum value

PKY3	=	5.0000e-001	$Variation of Kfy/Fznom with camber
PHY1	=	-5.0000e-003	$Horizontal shift Shy at Fznom
PHY2	=	-5.0000e-003	$Variation of shift Shy with load
PHY3	=	-5.0000e-004	$Variation of shift Shy with camber
PVY1	=	-5.0000e-003	$Vertical shift in Svy/Fz at Fznom
PVY2	=	5.0000e-003	$Variation of shift Svy/Fz with load
PVY3	=	-5.0000e-001	$Variation of shift Svy/Fz with camber
PVY4	=	5.0000e-001	$Variation of shift Svy/Fz with camber and load
RBY1	=	0.0000e+000	$Slope factor for combined Fy reduction
RBY2	=	0.0000e+000	$Variation of slope Fy reduction with alpha
RBY3	=	0.0000e+000	$Shift term for alpha in slope Fy reduction
RCY1	=	0.0000e+000	$Shape factor for combined Fy reduction
RHY1	=	0.0000e+000	$Shift factor for combined Fy reduction
RVY1	=	0.0000e+000	$Kappa induced side force Svyk/Muy*Fz at Fznom
RVY2	=	0.0000e+000	$Variation of Svyk/Muy*Fz with load
RVY3	=	0.0000e+000	$Variation of Svyk/Muy*Fz with camber
RVY4	=	0.0000e+000	$Variation of Svyk/Muy*Fz with alpha
RVY5	=	0.0000e+000	$Variation of Svyk/Muy*Fz with kappa
RVY6	=	0.0000e+000	$Variation of Svyk/Muy*Fz with atan(kappa)
PTY1	=	0.0000e+000	$Peak value of relaxation length SigAlp0/R0
PTY2	=	0.0000e+000	$Value of Fz/Fznom where SigAlp0 is extreme

$---rolling resistance

[ROLLING_COEFFICIENTS]

QSY1	=	0.0000e+000	$Rolling resistance torque coefficient
QSY2	=	0.0000e+000	$Rolling resistance torque depending on Fx

$---aligning

[ALIGNING_COEFFICIENTS]

QBZ1	=	5.0000e+001	$Trail slope factor for trail Bpt at Fznom
QBZ2	=	-5.0000e+000	$Variation of slope Bpt with load
QBZ3	=	-5.0000e+000	$Variation of slope Bpt with load squared
QBZ4	=	5.0000e-001	$Variation of slope Bpt with camber
QBZ5	=	5.0000e-003	$Variation of slope Bpt with absolute camber
QBZ9	=	5.0000e-001	$Slope factor Br of residual torque Mzr
QCZ1	=	5.0000e+000	$Shape factor Cpt for pneumatic trail
QDZ1	=	5.0000e-002	$Peak trail Dpt" = Dpt*(Fz/Fznom*R0)
QDZ2	=	5.0000e-004	$Variation of peak Dpt" with load
QDZ3	=	-5.0000e-001	$Variation of peak Dpt" with camber
QDZ4	=	5.0000e-001	$Variation of peak Dpt" with camber squared
QDZ6	=	-5.0000e-003	$Peak residual torque Dmr" = Dmr/(Fz*R0)
QDZ7	=	-5.0000e-004	$Variation of peak factor Dmr" with load
QDZ8	=	-5.0000e-002	$Variation of peak factor Dmr" with camber
QDZ9	=	5.0000e-002	$Var. of peak factor Dmr" with camber and load
QEZ1	=	-5.0000e+000	$Trail curvature Ept at Fznom
QEZ2	=	5.0000e+000	$Variation of curvature Ept with load
QEZ3	=	-5.0000e-001	$Variation of curvature Ept with load squared
QEZ4	=	-5.0000e-001	$Variation of curvature Ept with sign of Alpha-t
QEZ5	=	5.0000e-001	$Variation of Ept with camber and sign Alpha-t
QHZ1	=	-5.0000e-002	$Trail horizontal shift Sht at Fznom
QHZ2	=	-5.0000e-003	$Variation of shift Sht with load
QHZ3	=	5.0000e-002	$Variation of shift Sht with camber
QHZ4	=	5.0000e-001	$Variation of shift Sht with camber and load
SSZ1	=	0.0000e+000	$Nominal value of s/R0: effect of Fx on Mz
SSZ2	=	0.0000e+000	$Variation of distance s/R0 with Fy/Fznom
SSZ3	=	0.0000e+000	$Variation of distance s/R0 with camber
SSZ4	=	0.0000e+000	$Variation of distance s/R0 with load and camber
QTZ1	=	0.0000e+000	$Gyration torque constant
MBELT	=	0.0000e+000	$Belt mass of the wheelcurvature Efx while driving

APPENDIX B: MAGIC FORMULA EQUATIONS [1,2]

Longitudinal force Fx

$$F_x = (D_x \sin[C_x \arctan\{B_x\kappa_x - E_x(B_x\kappa_x - \arctan(B_x\kappa_x))\}] + SV_x)\cdot G_{x\alpha}$$

Where

$\kappa_x = \kappa + SH_x$
$C_x = P_{Cx1}\lambda_{Cx}$
$D_x = \mu_x F_z$
$\mu_x = (P_{Dx1} + P_{Dx2}df_z)\left(1 - P_{Dx3}\gamma^2\right)\lambda_{\mu_x}$
$E_x = (P_{Ex1} + P_{Ex2}df_z + P_{Ex3}df_z^2)\left(1 - P_{Ex4}sgn(\kappa_x)\right)\lambda_{Ex}$
$K_{x\kappa} = (P_{Kx1} + P_{Kx2}df_z)\exp(P_{Kx3}df_z)F_z\lambda_{Kx\kappa}$
$B_x = \dfrac{K_{x\kappa}}{C_x D_x}$
$S_{Hx} = (P_{Hx1} + P_{Hx2}df_z)\lambda_{Hx}$
$S_{Vx} = F_z(P_{Vx1} + P_{Vx2}df_z)\lambda_{Vx}\lambda_{\mu_x}$

Combined Slip:

$$G_{x\alpha} = \frac{\cos[C_{x\alpha}\arctan\{B_{x\alpha}\alpha_s - E_{x\alpha}(B_{x\alpha}\alpha_s - \arctan(B_{x\alpha}\alpha_s))\}]}{\cos[C_{x\alpha}\arctan\{B_{x\alpha}S_{Hx\alpha} - E_{x\alpha}(B_{x\alpha}S_{Hx\alpha} - \arctan(B_{x\alpha}S_{Hx\alpha}))\}]}$$

Where

$\alpha_s = \alpha_F + S_{Hx\alpha}$
$B_{x\alpha} = (r_{Bx1} + r_{Bx3}\gamma^2)\cos\{\arctan[r_{Bx2}\kappa]\}\lambda_{x\alpha}$
$C_{x\alpha} = r_{Cx1}$
$E_{x\alpha} = r_{Ex1} + r_{Ex2}df_z$
$S_{Hx\alpha} = r_{Hx1}$

For pure slip, $G_{x\alpha} = 1$.

Lateral force Fy

$$F_y = G_{y\kappa}F_{yp} + S_{Vy\kappa}$$

Where

$F_{yp} = (D_y \sin[C_y \arctan\{B_y\alpha_y - E_y(B_y\alpha_y - \arctan(B_y\alpha_y))\}] + S_{Vy})$
$\alpha_y = \alpha_F + S_{Hy}$
$C_y = P_{Cy1}\lambda_{Cy}$
$D_y = \mu_y F_z$
$\mu_y = (P_{Dy1} + P_{Dy2}df_z)\left(1 - P_{Dy3}\gamma^2\right)\lambda_{\mu_y}$
$E_y = (P_{Ey1} + P_{Ey2}df_z)\left(1 + P_{Ey5}\gamma^2 - (P_{Ey3} + P_{Ey4})sgn(\alpha_y)\right)\lambda_{Ey}$

$$K_{y\alpha} = P_{Ky1} F_{z0} \sin\left[P_{Ky4} \arctan\left\{\frac{F_z}{(P_{Ky2} + P_{Ky5}\gamma^2) F_{z0}}\right\}\right] (1 - P_{Ky3}|\gamma|) \lambda_{Ky\alpha}$$
$$B_y = \frac{K_{y\alpha}}{C_y D_y}$$
$$S_{Hy} = S_{Hy0} + S_{Hy\gamma}$$
$$S_{Hy0} = (P_{Hy1} + P_{Hy2} df_z) \lambda_{Hy}$$
$$S_{Hy\gamma} = \frac{K_{y\gamma} - S_{Vy\gamma}}{K_{y\alpha}}$$
$$S_{Vy0} = F_z (P_{Vy1} + P_{Vy2} df_z) \lambda_{Vy} \lambda_{\mu_y}$$
$$S_{Vy\gamma} = F_z (P_{Vy3} + P_{Vy4} df_z) \gamma \lambda_{Ky\gamma} \lambda_{\mu_y}$$

Combined Slip:

$$S_{Vy\kappa} = D_{Vy\kappa} \sin(r_{Vy5} \arctan(r_{Vy6} \kappa)) \lambda_{Vy\kappa}$$
$$D_{Vy\kappa} = \mu_y F_z (r_{Vy1} + r_{Vy2} df_z + r_{Vy3} \gamma) \cos(\arctan(r_{Vy4} \alpha_F))$$

$$G_{y\kappa} = \frac{\cos\left[C_{y\kappa} \arctan\left\{B_{y\kappa}\kappa_s - E_{y\kappa}\left(B_{y\kappa}\kappa_s - \arctan(B_{y\kappa}\kappa_s)\right)\right\}\right]}{\cos[C_{y\kappa} \arctan\{B_{y\kappa} S_{Hy\kappa} - E_{y\kappa}(B_{y\kappa} S_{Hy\kappa} - \arctan(B_{y\kappa} S_{Hy\kappa}))\}]}$$

Where

$$\kappa_s = \kappa_F + S_{Hx\kappa}$$
$$B_{y\kappa} = (r_{By1} + r_{By4}\gamma^2) \cos\{\arctan[r_{Bx2}(\alpha - r_{By3})]\} \lambda_{y\kappa}$$
$$C_{y\kappa} = r_{Cy1}$$
$$E_{y\kappa} = r_{Ey1} + r_{Ey2} df_z$$
$$S_{Hy\kappa} = r_{Hy1} + r_{Hy2} df_z$$

For pure slip, $S_{Vy\kappa} = 0$, $G_{y\kappa} = 1$.

Overturning moment Mx

$$M_x = R_0 F_z \lambda_{Mx} \left(Q_{Sx1} \lambda_{Mx} - Q_{Sx2} \gamma + \frac{Q_{Sx3} F_y}{F_{z0}}\right)$$

Rolling resistance moment My

For tire data where FITTYP is equal to 5:

$$M_y = R_0 (S_{Vx} + K_x S_{Hx})$$

Otherwise:

$$M_y = -R_0 F_{z0} \lambda_{My} \left\{\left(Q_{Sy1} + \frac{Q_{Sy2} F_x}{F_{z0}} + Q_{Sy3} \left|\frac{V_x}{V_{ref}}\right|\right) + Q_{Sy4} \left(\frac{V_x}{V_{ref}}\right)^4\right\}$$

Self aligning moment Mz

$$M_z = -t \cdot F_{y0} + M_{zr}$$

Where

$S_{Ht} = Q_{Hz1} + Q_{Hz2} df_z + (Q_{Hz3} + Q_{Hz4} df_z)\lambda_z$

$\alpha_t = \alpha + S_{Ht}$

$t = D_t \cos\left(C_t \arctan\left(B_t\alpha_t - E_t(B_t\alpha_t - \arctan(B_t\alpha_t))\right)\right) \cdot \cos(\alpha)$

$\alpha_r = \alpha + S_{Hf}$

$M_{zr} = D_r \cos\left(C_r \arctan(B_r\alpha_r)\right) \cdot \cos(\alpha)$

$\gamma_z = \gamma\, \lambda_{\gamma z}$

$B_t = (Q_{Bz1} + Q_{Bz2} df_z + Q_{Bz3} df_z^2)(1 + Q_{Bz4}\, \gamma_z + Q_{Bz5}|\gamma_z|)\lambda_{Ky}/\lambda_{\mu y}$

$C_t = Q_{Cz1}$

$D_t = F_z(Q_{Dz1} + Q_{Dz2} df_z)(1 + Q_{Dz3}\, \gamma_z + Q_{Dz4}\, \gamma_z^2)(R_0/F_{z0})\ \lambda_t$

$E_t = (Q_{Ez1} + Q_{Ez2} df_z + Q_{Ez3} df_z^2)\left(1 + (Q_{Ez4} + Q_{Ez5}\gamma_z) \cdot \frac{2}{\pi} \cdot \arctan(B_t C_t \alpha_t)\right)$ with $E_t \leq 1$

$B_r = Q_{Bz9} \frac{\lambda_{Ky}}{\lambda_{\mu y}} + Q_{Bz10} B_y C_y$

$C_r = 1$

$D_r = F_z\left((Q_{Dz6} + Q_{Dz7} df_z)\lambda_r + (Q_{Dz8} + Q_{Dz9} df_z)\lambda_z\right)R_0\lambda_{\mu y}$

APPENDIX C: MATLAB FUNCTION FILES

********** Fx.m ************

```
function [Fx] = Fx(kappa,Fz,gamma,filename,varargin)
%Fx Calculates the Longitudinal force
%   Fx calculates the longitudinal force in the x or y-direction using the
%   tire parameters in cell array S.
%
%   Input parameters:
%   kappa   Longitudinal slip
%   Fz      Force in the vertical direction
%   gamma   Inclination angle
%
%   Optional parameters:
%   Name    Values          Description
%   alpha   ALPMIN:ALPMAX   value of alpha for computing combined slip
%
%   Example: Fx(0.05, 9000, 0.1, 0.2)
%       alpha is set equal to 0.2
%   The equation is as follows:
%
%   F = (D*sin[C*arctan{B*ka - E(B*ka - arctan(Bx*ka))}] + SVx)*Gx

if nargin > 4
    % Combined slip
    alpha = varargin{1};
    G = fG( kappa, alpha, Fz, gamma, true, filename);
else % Pure slip
    G = 1;
end

Fx = G.*F(kappa, Fz, gamma, true, filename);
end
```

*************** Fy.m ***************

```
function [ Fy ] = Fy(alpha,Fz,gamma,filename, varargin)
%Fy calculates the lateral force
%   Using the tire parameters, Fy calculates the lateral force using the
%   following equation:
%
%   Fy = Gyk*Fyp + SVyk
%
%   Input parameters:
%   alpha   Slip angle
%   Fz      Force in the vertical direction
%   gamma   Inclination angle
%
%   Optional parameters:
%   Name    Values          Description
%   kappa   CAMMIN:CAMMAX   value of kappa for computing combined slip
%
%   Example: Fy(0.05, 9000, 0.1, 0.2)
%
%   If no optional arguments are given, the force is calculated for pure
%   slip where SVyk = 0 and Gyk = 1.

if nargin > 4
    kappa = varargin{1};
    Gyk = fG( kappa, alpha, Fz, gamma, false, filename);

    S = ImportTireData(filename);
```

```
    FNOMIN = gvar('FNOMIN',S);
    dfz = (Fz - FNOMIN)./Fz;

    % Calculate Muy
    Muy = fMu(gvar('PDY1',S), gvar('PDY2',S), gvar('PDY3',S), gamma, gvar('LMUY',S),
dfz);

    RVY1 = gvar('RVY1',S);
    RVY2 = gvar('RVY2',S);
    RVY3 = gvar('RVY3',S);
    RVY4 = gvar('RVY4',S);
    RVY5 = gvar('RVY5',S);
    RVY6 = gvar('RVY6',S);

    DVyk = Muy.*Fz.*(RVY1 + RVY2.*dfz + RVY3.*gamma).*cos(atan(RVY4.*alpha));

    SVyk = DVyk.*sin(RVY5.*atan(RVY6.*kappa));
else
    Gyk = 1;
    SVyk = 0;
end

Fy = F(alpha,Fz,gamma,false,filename).*Gyk + SVyk;

end

function [Mu] = fMu(PDX1, PDX2, PDX3, gamma, LMux, dfz)
% Inputs:
%  PDX1   Longitudinal friction Mux at Fznom
%  PDX2   Variation of friction Mux with load
% Ouput:
% Mu    Friction coefficient
% Mux = (PDX1 + PDX2*dfz)(1 - PDX3*gamma^2)*LMux
% **Removed:(1 + PPX3*dpi + PPX4*dpi^2)

Mu = (PDX1 + PDX2.*dfz).*(1 - PDX3.*gamma.^2).*LMux;
%**(1 + PPX3*dpi + PPX4*dpi^2)
end
```

********* F.m ************

```
function [ F ] = F(ka, Fz, gamma, isX, filename)
%F Calculates the magic formula for Fx or Fy
%   F calculates the force in the x or y-direction using the
%   tire parameters in cell array S.
%
%   Input parameters:
%   ka      Kappa (longitudinal slip) or alpha (slip angle)
%   Fz      Force in the vertical direction
%   gamma   Inclination angle
%   isX     Force direction flag: True = Calculate Fx, False = Calculate Fy
%
%   The equation is as follows:
%
%   F = (D*sin[C*arctan{B*ka - E(B*kax - arctan(Bx*kx))}] + SVx)*Gx
%
%   where
%   ka is either kappa or alpha
%   kappa: kx = k + SHx
%
```

```
%   C = PCX1*LCx
%   D = Mu*Fz
%
%   Mu = (PDX1 + PDX2*dfz)(1 - PDX3*gamma^2)*LMux
%       **Removed: (1 + PPX3*dpi + PPX4*dpi^2)
%
%   E = (PEX1 + PEX2*dfz + PEX3*dfz^2)(1 - PEX4*sgn(kx))*LEx
%   Kxk = (PKX1 + PKX2*dfz)exp(PKX3*dfz)*Fz*LKxk
%       **Removed: (1 + PPX1*dpi + PPX2*dpi^2)
%
%   B = Kxk/(Cx*Dx)
%   SH = (PHX1 + PHX2*dfz)*LHx
%   SV = (PVX1 + PVX2*dfz)*Fz*LVx*LMux

% Initialize S
S = ImportTireData(filename);

% Calculates dfz which is the dimensionless increment of vertical force Fz
% dfz = (Fz - Fz0)/Fz     where Fz0 = FNOMIN
FNOMIN = gvar('FNOMIN',S);
dfz = (Fz - FNOMIN)./Fz;

if isX == true %&& isValid == true
% Mux = (PDX1 + PDX2*dfz)(1 - PDX3*gamma^2)*LMux
        % **Removed:(1 + PPX3*dpi + PPX4*dpi^2)
PDX3 = 0;
% replaced PDX3 with 0 because PDX3 is not found in file
Mux = fMu(gvar('PDX1',S), gvar('PDX2',S), PDX3, gamma, gvar('LMUX',S), dfz);

% Calculate Dx
D = fD(Mux, Fz);

% Calculate Cx
C = fC(gvar('PCX1',S), gvar('LCX',S));

% Calculate Kxk
%   Kxk = (PKX1 + PKX2*dfz)exp(PKX3*dfz)*Fz*LKxk
%       **Removed: (1 + PPX1*dpi + PPX2*dpi^2)
PKX1 = gvar('PKX1',S);
PKX2 = gvar('PKX2',S);
PKX3 = gvar('PKX3',S);
LKxk = gvar('LKX',S);
Kxk = (PKX1 + PKX2.*dfz()).*exp(PKX3.*dfz()).*Fz.*LKxk;

% Calculate Bx
B = fB(Kxk, C, D);

% Calculate SHx
SH = fSH(gvar('PHX1',S),gvar('PHX2',S),gvar('LHX',S),dfz);

% Correct kappa
ka = ka + SH;

E = fEx(gvar('PEX1',S), gvar('PEX2',S), gvar('PEX3',S), dfz,
gvar('PEX4',S),ka,gvar('LEX',S));

SV = fSV(gvar('PVX1',S),gvar('PVX2',S),dfz,Fz,gvar('LVX',S),gvar('LMUX',S));

else %calculate y force
% Calculate Muy
Muy = fMu(gvar('PDY1',S), gvar('PDY2',S), gvar('PDY3',S), gamma, gvar('LMUY',S), dfz);
```

```
% Calculate Dy
D = fD(Muy, Fz);

% Calculate Cy
C = fC(gvar('PCY1',S), gvar('LCY',S));

% Calculate Kya
%   Kya = PKY1*FNOMIN*sin(PKY4*arctan(Fz/(PKY2 +
%   PKY5*gamma^2)/FNOMIN))*(1 - PKY3*abs(gamma))*LKya
% ** removed (1 + PPY1*dpi) twice
PKY1 = gvar('PKY1',S);
%FNOMIN already initiated
PKY4 = 2; %gvar('PKY4',S); *** not in imput data
PKY5 = 0; %gvar('PKY5',S); *** not in imput data
PKY2 = gvar('PKY2',S);
PKY3 = gvar('PKY3',S);
LFZO = gvar('LFZO',S);
LKY = gvar('LKY',S);
Kya = PKY1.*FNOMIN.*sin(PKY4.*atan(Fz./(PKY2+PKY5.*gamma.^2)./FNOMIN./LFZO)).*(1 -
PKY3.*abs(gamma)).*LKY;

% Calculate By
B = fB(Kya, C, D);

% Calculate Kyg
PVY3 = gvar('PVY3',S);
PVY4 = gvar('PVY4',S);
% ** removed (1 + PPY5*dpi)
Kyg = (PVY3 + PVY4.*dfz).*Fz;

% Calculate SHy
SHy0 = fSH(gvar('PHY1',S),gvar('PHY2',S),gvar('LHY',S),dfz);

SVyg = Fz.*(PVY3 + PVY4.*dfz).*gamma.*LKY.*gvar('LMUY',S);

SHyg = (Kyg.*gamma - SVyg)./Kya;

SH = SHy0 + SHyg;

% Calculate SVy
SVy0 = fSV(gvar('PVY1',S),gvar('PVY2',S),dfz,Fz, gvar('LVY ',S),gvar('LMUY',S));

SV = SVy0 + SVyg;

% Correct alpha
ka = ka + SH;

E = fEy(gvar('PEY1',S), gvar('PEY2',S), gvar('PEY3',S), dfz,
gvar('PEY4',S),ka,gvar('LEY',S),gamma);

end

% Calculate Fx
F = (D.*sin(C.*atan(B.*ka - E.*(B.*ka - atan(B.*ka))))+SV);
end

% Calculate Mu
function [Mu] = fMu(PDX1, PDX2, PDX3, gamma, LMux, dfz)
% Inputs:
%  PDX1   Longitudinal friction Mux at Fznom
```

```
%  PDX2   Variation of friction Mux with load
% Ouput:
% Mu    Friction coefficient
% Mux = (PDX1 + PDX2*dfz)(1 - PDX3*gamma^2)*LMux
% **Removed:(1 + PPX3*dpi + PPX4*dpi^2)

Mu = (PDX1 + PDX2.*dfz).*(1 - PDX3.*gamma.^2).*LMux;
%**(1 + PPX3*dpi + PPX4*dpi^2)
end

% Calculate D
function [D] = fD(Mu, Fz)
% Inputs:
% Mu    Friction coefficient
% Fz    Vertical reaction force
% Output:
% D     Peak value
D = Mu.*Fz;
end

% Calculate C
function [C] = fC(PCX1, LCx)
% Inputs:
%  PCX1    Shape factor Cfx for longitudinal force
%  LCx     Scale factor of Fx shape factor
% Ouput:
%  C
C = PCX1*LCx;
end

% Calculate B
function [B] = fB(K, C, D)
% Inputs:
%   K
%   C
%   D
% Ouput:
%   B
B = K./(C.*D);
end

% Calculate SH
function [SH] = fSH(PHX1, PHX2, LHx,dfz)
% Inputs:
%   PHX1    Horizontal shift Shx at Fznom
%   PHX2    Variation of shift Shx with load
%   LHX     Scale factor of Fx horizontal shift
%   dfz     Dimensionless increment of vertical force Fz
%
% Ouput:
%   SHx = (PHX1 + PHX2*dfz)*LHx

SH = (PHX1 + PHX2.*dfz()).*LHx;
end

function [E] = fEx(PEX1, PEX2, PEX3, dfz, PEX4,kappax,LEx)
% Inputs:
%   PEX1    Longitudinal curvature Efx at Fznom
%   PEX2    Variation of curvature Efx with load
%   PEX3    Variation of curvature Efx with load squared
%   PEX4    Factor in curvature Efx while driving
```

```
%   LEX     Scale factor of Fx curvature factor
%   kappax  ka scaled with SHx
%   dfz     Dimensionless increment of vertical force Fz
%
% Ouput:
%   E = (PEX1 + PEX2*dfz + PEX3*dfz^2)(1 - PEX4*sgn(kappax))*LEx

E = (PEX1 + PEX2.*dfz + PEX3.*dfz.^2).*(1 - PEX4.*sign(kappax)).*LEx;
end

function [E] = fEy(PEY1, PEY2, PEY3, dfz, PEY4,alphay,LEy,gamma)
% Inputs:
%   PEX1    Longitudinal curvature Efx at Fznom
%   PEX2    Variation of curvature Efx with load
%   PEX3    Variation of curvature Efx with load squared
%   PEX4    Factor in curvature Efx while driving
%   LEX     Scale factor of Fx curvature factor
%   kappax  ka scaled with SHx
%   dfz     Dimensionless increment of vertical force Fz
%
% Ouput:
%   E = (PEX1 + PEX2*dfz)(1 - (PEY3 + PEY4*gamma)*sgn(kappax))*LEx

E = (PEY1 + PEY2.*dfz).*(1 - (PEY3 + PEY4.*gamma).*sign(alphay)).*LEy;
if E > ones(size(E))
    warning(['Ey = ' E ' which is greater than or equal to 1']);
end
end

function [SV] = fSV(PVX1,PVX2,dfz,Fz,LVx,LMux)
% Inputs:
%   PVX1    Vertical shift Svx/Fz at Fznom
%   PVX2    Variation of shift Svx/Fz with load
%   LVx     Scale factor of Fx vertical shift
%   LMux    Scale factor of Fx peak friction coefficient
%   Fz      ka scaled with SHx
%   dfz     Dimensionless increment of vertical force Fz
%
% Ouput:
%   SV = (PVX1 + PVX2*dfz)*Fz*LVx*LMux
SV = Fz.*(PVX1 + PVX2.*dfz).*LVx.*LMux;
end
```

********** MomentCalc.m *********

```
function [ Mx,My,Mz ] = MomentCalc(fn, Fz, Fx, Fy, gamma, alpha, varargin)
%MOMENTCALC Calculates the moment in the x, y, and z directions
%   MomentCalc calculates the overturning moment, Mx; rolling resistance
%   moment, My; and the self aligning moment, Mz.
%
%   Input parameters:
%   fn      Tire data file name
%   Fz      Vertical force
%   Fx      Longitudinal force
%   Fy      Lateral force
%   gamma   Inclination angle
%   varargin  Vx, velocity in x direction if FITTYP does not equal 5
%
%   Overturning Moment, Mx
%   Mx = R0*Fz*LMx*(QSX1*LVMx - QSX2*gamma + QSX3*Fy/FNOMIN)
%
```

```
%   Rolling Resistance moment, My
%   if FITTYP ~= 5
%       My = -R0*FNOMIN*LMy*(QSY1 + QSY2*Fx/FNOMIN + QSY3*abs(Vx/LONGVL) +
%       QSY4*(Vx/LONGVL)^4
%   else
%       My = R0*(SVx + Kx*SHx)
%
%   Self aligning moment, Mz
%   Mz = -t * Fy0 + Mzr
% where
%   SHt = QHZ1 + QHZ2*dfz + (QHZ3 + QHZ4*dfz)*gz
%   at = alpha + SHt
%   t = Dt*cos(Ct*atan(Bt*at - Et*(Bt*at - atan(Bt*at))))*cos(alpha)
%   ar = alpha + SHf
%   Mzr = Dr*cos(Cr*atan(Br*ar))*cos(alpha)
%   gz = gamma*LGAZ
%   Bt = (QBZ1 + QBZ2*dfz + QBZ3*dfz^2)*(1 + QBZ4*gz +
%                   QBZ5*abs(gz))*LKY/LMuy
%   Ct = QCZ1
%   Dt = Fz*(QDZ1 + QDZ2*dfz)*(1 + QDZ3*gz + QDZ4*gz^2)*R0/FZNOMIN*LTR
%   Et = (QEZ1 + QEZ2*dfz + QEZ3*dfz^2)*(1 + (QEZ4 +
%           QEZ5*gz)*2/pi*atan(Bt*Ct*at)) with Et <= 1
%   Br = QBZ9*LKY/LMUY + QBZ10*By*Cy
%   Cr = 1
%   Dr = Fz*((QDZ6 + QDZ7*dfz)*LRES + (QDZ8 + QDZ9*dfz)*gz)*R0*LMuy
S = ImportTireData(fn);

%   Overturning Moment, Mx
R0 = gvar('UNLOADED_RADIUS',S);
LMx = gvar('LMX',S);
QSX1 = gvar('QSX1',S);
QSX2 = gvar('QSX2',S);
QSX3 = gvar('QSX3',S);
FNOMIN = gvar('FNOMIN',S);
LVMx = 1;

Mx = R0.*Fz.*LMx.*(QSX1.*LVMx - QSX2.*gamma + QSX3.*Fy./FNOMIN);

%   Rolling Resistance moment, My
QSY1 = gvar('QSY1',S);
QSY2 = gvar('QSY2',S);
dfz = (Fz - FNOMIN)./Fz;
if gvar('FITTYP',S) ~= 5
    Vx = varargin{1};
    LMy = gvar('LMY',S);
    QSY3 = gvar('QSY3',S);
    LONGVL = gvar('LONGVL',S);

    My = -R0.*FNOMIN.*LMy.*(QSY1 + QSY2.*Fx./FNOMIN + QSY3.*abs(Vx./LONGVL)...
        + QSY4.*(Vx./LONGVL)^4);
elseif QSY1 == 0 && QSY2 == 0
    PKX1 = gvar('PKX1',S);
    PKX2 = gvar('PKX2',S);
    PKX3 = gvar('PKX3',S);
    LKx = gvar('LKX',S);
    Kx = (PKX1 + PKX2.*dfz).*exp(PKX3.*dfz).*Fz.*LKx;
    SVx = fSV(gvar('PVX1',S),gvar('PVX2',S),dfz,Fz,gvar('LVX',S),gvar('LMUX',S));
    SHx = fSH(gvar('PHX1',S),gvar('PHX2',S),gvar('LHX',S),dfz);

    My = R0.*(SVx + Kx.*SHx);
end
```

```
%   Self aligning moment, Mz
QHZ1 = gvar('QHZ1',S);
QHZ2 = gvar('QHZ2',S);
QHZ3 = gvar('QHZ3',S);
QHZ4 = gvar('QHZ4',S);
LGAZ = gvar('LGAZ',S);

gz = gamma.*LGAZ;
SHt = QHZ1 + QHZ2.*dfz + (QHZ3 + QHZ4.*dfz).*gz;
at = alpha + SHt;

QBZ1 = gvar('QBZ1',S);
QBZ2 = gvar('QBZ2',S);
QBZ3 = gvar('QBZ3',S);
QBZ4 = gvar('QBZ4',S);
QBZ5 = gvar('QBZ5',S);
LKY  = gvar('LKY',S);
LMUY = gvar('LMUY',S);

Bt = (QBZ1 + QBZ2.*dfz + QBZ3.*dfz.^2).*(1 + QBZ4.*gz + QBZ5.*abs(gz)).*LKY./LMUY;

QCZ1 = gvar('QCZ1',S);
QDZ1 = gvar('QDZ1',S);
QDZ2 = gvar('QDZ2',S);
QDZ3 = gvar('QDZ3',S);
QDZ4 = gvar('QDZ4',S);
LTR  = gvar('LTR',S);

Ct = QCZ1;
Dt = Fz.*(QDZ1 + QDZ2.*dfz).*(1 + QDZ3.*gz + QDZ4.*gz.^2).*R0./FNOMIN.*LTR;

QEZ1 = gvar('QEZ1',S);
QEZ2 = gvar('QEZ2',S);
QEZ3 = gvar('QEZ3',S);
QEZ4 = gvar('QEZ4',S);
QEZ5 = gvar('QEZ5',S);

Et = (QEZ1 + QEZ2.*dfz + QEZ3.*dfz.^2).*(1 + (QEZ4 + ...
      QEZ5.*gz).*2./pi.*atan(Bt.*Ct.*at));
if Et > ones(size(Et))
    warning(['Et = ' Et ' which is greater than one.']);
end

t = Dt.*cos(Ct.*atan(Bt.*at - Et.*(Bt.*at - atan(Bt.*at)))).*cos(alpha);


%Residual moment, Mzr
SHy = fSH(gvar('PHY1',S),gvar('PHY2',S),gvar('LHY',S),dfz);
SVy = fSV(gvar('PVY1',S),gvar('PVY2',S),dfz,Fz, gvar('LVY ',S),gvar('LMUY',S));
PKY1 = gvar('PKY1',S);
PKY4 = 2; %gvar('PKY4',S); *** not in imput data
PKY5 = 0; %gvar('PKY5',S); *** not in imput data
PKY2 = gvar('PKY2',S);
PKY3 = gvar('PKY3',S);
LFZO = gvar('LFZO',S);
Ky = PKY1.*FNOMIN.*sin(PKY4.*atan(Fz./(PKY2+PKY5.*gamma.^2)./FNOMIN./LFZO)).*(1 -
PKY3.*abs(gamma)).*LKY;

SHf = SHy + SVy./Ky;
ar = alpha + SHf;
```

```
QBZ9 = gvar('QBZ9',S);
QBZ10 = 1; %gvar('QBZ10',S); *** not found in tire data
QDZ6 = gvar('QDZ6',S);
QDZ7 = gvar('QDZ7',S);
LRES = gvar('LRES',S);
QDZ8 = gvar('QDZ8',S);
QDZ9 = gvar('QDZ9',S);

%****
% Calculate Muy
Muy = fMu(gvar('PDY1',S), gvar('PDY2',S), gvar('PDY3',S), gamma, gvar('LMUY',S), dfz);

% Calculate Dy
Dy = fD(Muy, Fz);

% Calculate Cy
Cy = fC(gvar('PCY1',S), gvar('LCY',S));

% Calculate By
By = fB(Ky, Cy, Dy);
%****

Br = QBZ9.*LKY./LMUY + QBZ10.*By.*Cy;
Cr = 1;
Dr = Fz.*((QDZ6 + QDZ7.*dfz).*LRES + (QDZ8 + QDZ9.*dfz).*gz).*R0.*LMUY;
Mzr = Dr.*cos(Cr.*atan(Br.*ar)).*cos(alpha);

Mz = -t.*Fy + Mzr;

end

function [SV] = fSV(PVX1,PVX2,dfz,Fz,LVx,LMux)
% Inputs:
%   PVX1    Vertical shift Svx/Fz at Fznom
%   PVX2    Variation of shift Svx/Fz with load
%   LVx     Scale factor of Fx vertical shift
%   LMux    Scale factor of Fx peak friction coefficient
%   Fz      ka scaled with SHx
%   dfz     Dimensionless increment of vertical force Fz
%
% Ouput:
%   SV = (PVX1 + PVX2*dfz)*Fz*LVx*LMux
SV = Fz.*(PVX1 + PVX2.*dfz).*LVx.*LMux;
end

function [SH] = fSH(PHX1, PHX2, LHx,dfz)
% Inputs:
%   PHX1    Horizontal shift Shx at Fznom
%   PHX2    Variation of shift Shx with load
%   LHX     Scale factor of Fx horizontal shift
%   dfz     Dimensionless increment of vertical force Fz
%
% Ouput:
%   SHx = (PHX1 + PHX2*dfz)*LHx

SH = (PHX1 + PHX2.*dfz).*LHx;
end

% Calculate Mu
function [Mu] = fMu(PDX1, PDX2, PDX3, gamma, LMux, dfz)
```

```
% Inputs:
%
% Ouput:
% Mu    Friction coefficient
% Mux = (PDX1 + PDX2*dfz)(1 - PDX3*gamma^2)*LMux
% **Removed:(1 + PPX3*dpi + PPX4*dpi^2)

Mu = (PDX1 + PDX2.*dfz).*(1 - PDX3.*gamma.^2).*LMux;
%**(1 + PPX3*dpi + PPX4*dpi^2)
end

% Calculate D
function [D] = fD(Mu, Fz)
% Inputs:
% Mu    Friction coefficient
% Fz    Vertical reaction force
% Output:
% D     Peak value
D = Mu.*Fz;
end

% Calculate C
function [C] = fC(PCX1, LCx)
% Inputs:
%
% Ouput:
%
C = PCX1*LCx;
end

% Calculate B
function [B] = fB(K, C, D)
% Inputs:
%
% Ouput:
%
B = K./(C.*D);
end
```

**************** **ImportTireData.m** ********************

```
function [ C ] = ImportTireData(filename)
%ImportTireData
% ImportTireData( filename ) returns a cell array, C, which contains the
% variable names in the first cell and the data for each variable name.
% in the second cell.

fid = fopen(filename);

C = cell(1,2);
pos = 1;
sizeC = 0;

while ~feof(fid)
    [a, pos] = textscan(fid, '%s = %12f',pos, 'CommentStyle', '$');
    while size(a{1},1) <= 1
        [a,pos] = textscan(fid, '%s = %12f',pos, 'CommentStyle', '$');
    end

    for i = 1:size(a{2},1)
        C{1}(sizeC+i) = a{1}(i);
```

```
        C{2}(sizeC+i) = a{2}(i);
    end
    sizeC = size(C{1},2);
end

fclose(fid); % Close file

end
```

*********** fG.m **************

```
function [ G ] = fG( kappa, alpha, Fz, gamma, isX, filename)
%fG calculates the combined slip modification factor for the x or y direction
%   fG  uses the tire parameters as well as the inputs to compute the
%   combined slip factor G, which equation is shown below:
%
%   G = (cos(C*atan(B*ak - E*(B*alphaS - atan(B*alphaS)))))/
%       (cos(C*atan(B*SH - E*(B*SH - atan(B*SH)))))
%
%   where
%   alphaS = alphaF + SH
%   B = (RBX1 + RBX3*gamma^2)*cos(atan(RBX2*kappa))*LXA
%   C = RCX1
%   E = REX1 + REX2
%   SH = RHX1

S = ImportTireData(filename);

FNOMIN = gvar('FNOMIN',S);
dfz = (Fz - FNOMIN)./Fz;

if gvar('USE_MODE',S) ~= 4
    warning('Tire data file is not configured for combined slip. \n USE_MODE = %d:
Fx,Fy,Mx,My,Mz uncombined force/moment calculation', ...
        gvar('USE_MODE',S));
    G = 1;
else
    if isX
        RHX1 = gvar('RHX1',S);
        RBX1 = gvar('RBX1',S);
        RBX2 = gvar('RBX2',S);
        RBX3 = 0; %not in tire data gvar('RBX3',S);
        RCX1 = gvar('RCX1',S);
        REX1 = gvar('REX1',S);
        REX2 = gvar('REX2',S);
        LXA = gvar('LXAL',S);

        B = (RBX1 + RBX3.*gamma.^2).*cos(atan(RBX2.*kappa)).*LXA;
        SH = RHX1;
        akS = alpha + SH; %akS: alphaS = alphaF + SHxa
    else
        RHY1 = gvar('RHY1',S);
        RHY2 = gvar('RHY2',S);
        RBY1 = gvar('RBY1',S);
        RBY2 = gvar('RBY2',S);
        RBY3 = gvar('RBY3',S);
        RBY4 = 0; %not in tire data gvar('RBY4',S);
        RCX1 = gvar('RCY1',S);
        REX1 = gvar('REY1',S);
        REX2 = gvar('REY2',S);
        LYK = gvar('LYKA',S);
```

```
        B = (RBY1 + RBY4.*gamma.^2).*cos(atan(RBY2.*(alpha - RBY3))).*LYK;
        SH = RHY1 + RHY2.*dfz;
        akS = kappa + SH;  %akS: kappaS = kappa + SHyk
    end
    C = RCX1;
    E = REX1 + REX2.*dfz;

    G = (cos(C.*atan(B.*akS - E.*(B.*akS - atan(B.*akS))))./...
        (cos(C.*atan(B.*SH - E.*(B.*SH - atan(B.*SH)))));
end

end
```

*********** gvar.m *********

```
function [ B ] = gvar(varName, S )
%GVAR Gets data from S corresponding to varName
%   GVAR (varName as String, S as 1x2 Cell Array) searchs for an exact match of
%   varName in the first column of cell array S. If a match is found, GVAR returns
%   the value from the 2nd column of S with the same row index.

%   Date Last Revised: 6 Aug 10

x = strmatch(varName, S{1});

if x > 0
    B = S{2}(x);
else
    error(['The variable ' varName ' was not found in imported data.'])
end

end
```

********** Pacejka_Plot_Tool.m ***********

```
function varargout = Pacejka_Plot_Tool(varargin)
% PACEJKA_PLOT_TOOL M-file for Pacejka_Plot_GUI.fig
%      PACEJKA_PLOT_TOOL, by itself, creates a new PACEJKA_PLOT_TOOL or raises the
%      existing singleton*.
%
%      H = PACEJKA_PLOT_TOOL returns the handle to a new PACEJKA_PLOT_TOOL or the handle
%      to the existing singleton*.
%
%      PACEJKA_PLOT_TOOL('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in PACEJKA_PLOT_TOOL.M with the given input arguments.
%
%      PACEJKA_PLOT_TOOL('Property','Value',...) creates a new PACEJKA_PLOT_TOOL or
%      raises the existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before Pacejka_Plot_Tool_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to Pacejka_Plot_Tool_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help Pacejka_Plot_Tool

% Last Modified by GUIDE v2.5 14-Sep-2010 9:35:14
```

```
% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @Pacejka_Plot_Tool_OpeningFcn, ...
                   'gui_OutputFcn',  @Pacejka_Plot_Tool_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before Pacejka_Plot_GUI is made visible.
function Pacejka_Plot_Tool_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to Pacejka_Plot_GUI (see VARARGIN)

% Choose default command line output for Pacejka_Plot_GUI
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% Set up the initial plot - only do when we are invisible
% so window can get raised using Pacejka_Plot_GUI.
if strcmp(get(hObject,'Visible'),'off')
    % Find tire data files in current directory to populate list box
    initial_dir = pwd;
    load_listbox(initial_dir,handles)

    % Check if list was populated from current directory
    list_entries = get(handles.lstFn, 'String');
    if isempty(list_entries)
        % See if tire data is located in Tire Data folder
        tempPath = fliplr(pwd);
        [~, rem] = strtok(tempPath, '\');
        new_dir = [fliplr(rem) 'Tire Data\'];
        load_listbox(new_dir, handles);

        % If list is still empty, ask user for location
        list_entries = get(handles.lstFn, 'String');
        if isempty(list_entries)
            h = msgbox('Select a tire data file.','modal');
            uiwait(h);
            % Prompt user to choose file/folder where data is located
            btnChangePath_Callback(hObject, eventdata, handles);
        else
            % Tire data was found, plot selected data
            popPlotType_Callback(hObject, eventdata, handles);
        end
    else
```

```
        % Plot selected data
        popPlotType_Callback(hObject, eventdata, handles);
    end
end

degrees = radtodeg(str2double(get(handles.txtGamma, 'String')));
text_deg = sprintf('(%2.1f deg)',degrees);
set(handles.lblDeg, 'String',text_deg);


% --- Outputs from this function are returned to the command line.
function varargout = Pacejka_Plot_Tool_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;

% --- Executes on button press in btnChangePath.
function btnChangePath_Callback(hObject, eventdata, handles)
% hObject    handle to btnChangePath (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Gets a user-selected file using standard open dialog
[fn,path,FilterIndex] = uigetfile('*.tir','Select a tire data file');

% Load list box with *.tir files in selected folder
load_listbox(path, handles);

% Make the user-selected file become the selected file in the listbox
fn_index = strmatch(fn,get(handles.lstFn,'String'));
set(handles.lstFn, 'Value', fn_index);

% Plot selected file
popPlotType_Callback(hObject, eventdata, handles);



% --------------------------------------------------------------------
function FileMenu_Callback(hObject, eventdata, handles)
% hObject    handle to FileMenu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)


% --------------------------------------------------------------------
function OpenMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to OpenMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
file = uigetfile('*.fig');
if ~isequal(file, 0)
    open(file);
end

% --------------------------------------------------------------------
function PrintMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to PrintMenuItem (see GCBO)
```

```
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
printdlg(handles.figure1)

% --------------------------------------------------------------------
function CloseMenuItem_Callback(hObject, eventdata, handles)
% hObject    handle to CloseMenuItem (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
selection = questdlg(['Close ' get(handles.figure1,'Name') '?'],...
                     ['Close ' get(handles.figure1,'Name') '...'],...
                     'Yes','No','Yes');
if strcmp(selection,'No')
    return;
end

delete(handles.figure1)


function filename = getFn(handles)
% Get first selected value from list box
list_entries = get(handles.lstFn, 'String');
index_selected = get(handles.lstFn, 'Value');

dir_path = get(handles.txtPath, 'String');

filename = fullfile(dir_path,list_entries{index_selected(1)});



% --- Executes on selection change in popPlotType.
function popPlotType_Callback(hObject, eventdata, handles)
% hObject    handle to popPlotType (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns popPlotType contents as cell array
%        contents{get(hObject,'Value')} returns selected item from popPlotType

% Get selected value from popup box
popup_sel_index = get(handles.popPlotType, 'Value');

% Get selected values from list box
list_entries = get(handles.lstFn, 'String');
index_selected = get(handles.lstFn, 'Value');

% Get current path from text box
dir_path = get(handles.txtPath, 'String');

for i = 1:length(index_selected)
% Construct file name from path and list box selection
curName = list_entries{index_selected(i)};
fn = fullfile(dir_path,curName);

% Parse Fz string/s into double
Fz_text = get(handles.txtFz, 'String');
Fz = str2double(regexp(Fz_text, '[0-9.e+-\s]*[^;]','match'));

% Get gamma from text box and convert to double
gamma = str2double(get(handles.txtGamma, 'String'));
```

```
% Calculate kappa and alpha increments from user supplied max and min
kappa = calcIncrement(handles.txtKappaMin, handles.txtKappaMax, handles);
alpha = calcIncrement(handles.txtAlphaMin, handles.txtAlphaMax, handles);

% Number of steps
n = str2double(get(handles.txtStep, 'String'));

% Make input matrices the same dimensions (i.e. 1x3 and 30x1 to 3x30)
Fz = make_square(Fz, n, ';');
kappa = make_square(kappa, size(Fz,2), ' ');
alpha = make_square(alpha, size(Fz,2), ' ');

% Select pure or combined slip based on USE_MODE in tire file
S = ImportTireData(fn);
use_mode = gvar('USE_MODE',S);
if use_mode == 3
    % Calculate Fx and Fy for pure slip
    x = Fx(kappa,Fz,gamma,fn);
    y = Fy(alpha,Fz,gamma,fn);
elseif use_mode == 4
    % Calculate Fx and Fy for combined slip
    x = Fx(kappa,Fz,gamma,fn,alpha);
    y = Fy(alpha,Fz,gamma,fn,kappa);
else
    msgbox(['Unrecognized USE_MODE in tire data file. USE_MODE = ' use_mode])
end

% Calculate moments
[Mx, My, Mz] = MomentCalc(fn, Fz, x, y, gamma, alpha);


% Plot forces and moments based on user-selected plot type
xlabel_text = 'slip (% or rad)';
switch popup_sel_index
    case 1 % Fx and Fy
        plot1 = plot(kappa,x,'Color','b');
        hold on;
        plot2 = plot(alpha,y,'Color','g');
        xlabel(xlabel_text)
        ylabel('Force (N)')

        % Set line type and legend entries
        formatExpr = '%s (Fz = %2.3g kN)';
        for j = 1:size(Fz,2)
            if j == 2      % dashed
                set(plot1(j),'LineStyle','--')
                set(plot2(j),'LineStyle','--')
            elseif j == 3 % dotted
                set(plot1(j),'LineStyle',':')
                set(plot2(j),'LineStyle',':')
            elseif j == 4 % dash-dot
                set(plot1(j),'LineStyle','-.')
                set(plot2(j),'LineStyle','-.')
            end
            set(plot1(j),'DisplayName',sprintf(formatExpr,'Fx',Fz(1,j)/1000));
            set(plot2(j),'DisplayName',sprintf(formatExpr,'Fy',Fz(1,j)/1000));
        end

        legend(handles.axes1,'show');
    case 2 % Mx, My, and Mz
        plot1 = plot(alpha,Mx,'Color','r');
```

```
        hold on;
        plot2 = plot(alpha,My,'Color','m');
        plot3 = plot(alpha,Mz,'Color','c');

        xlabel(xlabel_text)
        ylabel('Moment (N m)')

        % Set line type and legend entries
        formatExpr = '%s (Fz = %2.3g kN)';
        for j = 1:size(Fz,2)
            if j == 2      % dashed
                set(plot1(j),'LineStyle','--')
                set(plot2(j),'LineStyle','--')
                set(plot3(j),'LineStyle','--')
            elseif j == 3  % dotted
                set(plot1(j),'LineStyle',':')
                set(plot2(j),'LineStyle',':')
                set(plot3(j),'LineStyle',':')
            elseif j == 4  % dash-dot
                set(plot1(j),'LineStyle','-.')
                set(plot2(j),'LineStyle','-.')
                set(plot3(j),'LineStyle','-.')
            end
            set(plot1(j),'DisplayName',sprintf(formatExpr,'Mx',Fz(1,j)/1000));
            set(plot2(j),'DisplayName',sprintf(formatExpr,'My',Fz(1,j)/1000));
            set(plot3(j),'DisplayName',sprintf(formatExpr,'Mz',Fz(1,j)/1000));
        end

        legend(handles.axes1,'show');
    case 3 % Fx only
        plot1 = plot(kappa,x,'Color','b');hold on;
        ylabel('Force (N)')
        xlabel(xlabel_text)

        % Set line type and legend entries
        formatExpr = 'Fz = %2.3g kN';
        for j = 1:size(Fz,2)
            if j == 2
                % dashed
                set(plot1(j),'LineStyle','--')
            elseif j == 3
                % dotted
                set(plot1(j),'LineStyle',':')
            elseif j == 4
                % dash-dot
                set(plot1(j),'LineStyle','-.')
            end
            set(plot1(j),'DisplayName',sprintf(formatExpr,Fz(1,j)/1000));
        end

        legend(handles.axes1,'show');
    case 4 % Fy only
        plot1 = plot(alpha,y,'Color','g');hold on;
        ylabel('Force (N)')
        xlabel(xlabel_text)

        % Set line type and legend entries
        formatExpr = 'Fz = %2.3g kN';
        for j = 1:size(Fz,2)
            if j == 2      % dashed
                set(plot1(j),'LineStyle','--')
```

```
        elseif j == 3  % dotted
            set(plot1(j),'LineStyle',':')
        elseif j == 4  % dash-dot
            set(plot1(j),'LineStyle','-.')
        end
        set(plot1(j),'DisplayName',sprintf(formatExpr,Fz(1,j)/1000));
    end
    legend(handles.axes1,'show');
case 5 % Mx only
    plot1 = plot(alpha,Mx,'Color','r');hold on;
    xlabel(xlabel_text)
    ylabel('Moment (N m)')

    % Set line type and legend entries
    formatExpr = 'Fz = %2.3g kN';
    for j = 1:size(Fz,2)
        if j == 2
            % dashed
            set(plot1(j),'LineStyle','--')
        elseif j == 3
            % dotted
            set(plot1(j),'LineStyle',':')
        elseif j == 4
            % dash-dot
            set(plot1(j),'LineStyle','-.')
        end
        set(plot1(j),'DisplayName',sprintf(formatExpr,Fz(1,j)/1000));
    end
    legend(handles.axes1,'show');
case 6 % My only
    plot1 = plot(alpha,My,'Color','m');hold on;
    xlabel(xlabel_text)
    ylabel('Moment (N m)')

    % Set line type and legend entries
    formatExpr = 'Fz = %2.3g kN';
    for j = 1:size(Fz,2)
        if j == 2      % dashed
            set(plot1(j),'LineStyle','--')
        elseif j == 3  % dotted
            set(plot1(j),'LineStyle',':')
        elseif j == 4  % dash-dot
            set(plot1(j),'LineStyle','-.')
        end
        set(plot1(j),'DisplayName',sprintf(formatExpr,Fz(1,j)/1000));
    end
    legend(handles.axes1,'show');
case 7 % Mz only
    plot1 = plot(alpha,Mz,'Color','c');hold on;
    xlabel(xlabel_text)
    ylabel('Moment (N m)')

    % Set line type and legend entries
    formatExpr = 'Fz = %2.3g kN';
    for j = 1:size(Fz,2)
        if j == 2        % dashed
            set(plot1(j),'LineStyle','--')
        elseif j == 3    % dotted
            set(plot1(j),'LineStyle',':')
        elseif j == 4    % dash-dot
            set(plot1(j),'LineStyle','-.')
```

```
        end
        set(plot1(j),'DisplayName',sprintf(formatExpr,Fz(1,j)/1000));
    end
    legend(handles.axes1,'show');
case 8 % All forces and moments
    hFxPlot = plot(kappa,x,'Color','b');
    hold on;
    hFyPlot = plot(alpha,y,'Color','g');
    hMxPlot = plot(alpha,Mx,'Color','r');
    hMyPlot = plot(alpha,My,'Color','m');
    hMzPlot = plot(alpha,Mz,'Color','c');

    xlabel(xlabel_text)
    ylabel('Force (N) / Moment (N m)')

    % Group together so multiple lines do not show on legend
    hFxGroup = hggroup;
    hFyGroup = hggroup;
    hMxGroup = hggroup;
    hMyGroup = hggroup;
    hMzGroup = hggroup;

    set(hFxPlot,'Parent',hFxGroup)
    set(hFyPlot,'Parent',hFyGroup)
    set(hMxPlot,'Parent',hMxGroup)
    set(hMyPlot,'Parent',hMyGroup)
    set(hMzPlot,'Parent',hMzGroup)

    % Include these hggroups in the legend:
    set(get(get(hFxGroup,'Annotation'),'LegendInformation'),...
        'IconDisplayStyle','on');
    set(get(get(hFyGroup,'Annotation'),'LegendInformation'),...
        'IconDisplayStyle','on');
    set(get(get(hMxGroup,'Annotation'),'LegendInformation'),...
        'IconDisplayStyle','on');
    set(get(get(hMyGroup,'Annotation'),'LegendInformation'),...
        'IconDisplayStyle','on');
    set(get(get(hMzGroup,'Annotation'),'LegendInformation'),...
        'IconDisplayStyle','on');
    % Show legend
    legend('Fx','Fy','Mx','My','Mz')


    % Change line size of multiple Fz plots
    for j = 1:size(Fz,2)
        if j == 2 % dashed
            set(hFxPlot(j),'LineStyle','--')
            set(hFyPlot(j),'LineStyle','--')
            set(hMxPlot(j),'LineStyle','--')
            set(hMyPlot(j),'LineStyle','--')
            set(hMzPlot(j),'LineStyle','--')
        elseif j == 3 % dotted
            set(hFxPlot(j),'LineStyle',':')
            set(hFyPlot(j),'LineStyle',':')
            set(hMxPlot(j),'LineStyle',':')
            set(hMyPlot(j),'LineStyle',':')
            set(hMzPlot(j),'LineStyle',':')
        elseif j == 4 % dash-dot
            set(hFxPlot(j),'LineStyle','-.')
            set(hFyPlot(j),'LineStyle','-.')
            set(hMxPlot(j),'LineStyle','-.')
```

```
                set(hMyPlot(j),'LineStyle','-.')
                set(hMzPlot(j),'LineStyle','-.')
            end
        end

end
end

hold off;


function trans_var = calcIncrement(hMin, hMax, handles)
% calcIncrement returns an array with the number of increments specified by
% txtStep (Resolution) between the text box of that handle hMin specifies
% and the text box of that handle hMax specifies.
% Input:
%   hMin        handle to textbox with minimum value
%   hMax        handle to textbox with maximum value
%   handles     structure with handles and user data (see GUIDATA)


% Get values for Min and Max
Min = str2double(get(hMin, 'String'));
Max = str2double(get(hMax,'String'));

% Number of steps
n = str2double(get(handles.txtStep, 'String'));

% Increment size
inc = (Max - Min)/(n-1);
increm_var = Min:inc:Max;

% Transposes array
trans_var = increm_var';

function square = make_square(unsquare, n, delim)
% make_square changes a 1x3 matrix to a nx3 matrix if delim = ';'
% make_square changes an 30x1 matrix to a 30xn matrix if delim = ' ' (or
% anything else)

square = [];
if delim == ';'
    for i = 1:n
        square = [square; unsquare];
    end
else % delim = ' '
    for i = 1:n
        square = [square unsquare];
    end
end


% --- Executes during object creation, after setting all properties.
function popPlotType_CreateFcn(hObject, eventdata, handles)
% hObject    handle to popPlotType (see GCBO)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
     set(hObject,'BackgroundColor','white');
end

% Set plot type options
```

```
set(hObject, 'String', {'Fx & Fy', 'Mx, My, & Mz', 'Fx', 'Fy', 'Mx', 'My', 'Mz', 'All'});


function txtLowB_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)
popPlotType_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function txtLowB_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtLowB (see GCBO)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

set(hObject, 'String', '-0.4');


function txtUpperB_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)
popPlotType_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function txtUpperB_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtUpperB (see GCBO)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

set(hObject, 'String', '0.4');



function txtStep_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)
popPlotType_Callback(hObject, eventdata, handles);


function txtStep_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtStep (see GCBO)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
% Set Default String
set(hObject, 'String', '30');


function lstFn_Callback(hObject, eventdata, handles)
% hObject    handle to lstFn (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Set automatically updated force if checked
if get(handles.chkUpdate, 'Value') == get(handles.chkUpdate, 'Max')
    S = ImportTireData(getFn(handles));
    FzMin = gvar('FZMIN',S);
    FzMax = gvar('FZMAX',S);
```

```
    FzAvg = (FzMax+FzMin)/2;
    FzRange = sprintf('%d;%d;%d',FzMin,FzAvg,FzMax);
    set(handles.txtFz, 'String', FzRange);
end
% Refresh plot
popPlotType_Callback(hObject, eventdata, handles);


% --- Executes during object creation, after setting all properties.
function lstFn_CreateFcn(hObject, eventdata, handles)
% hObject    handle to lstFn (see GCBO)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function load_listbox(dir_path, handles)
dir_struct = dir(fullfile(dir_path,'*.tir'));
[sorted_names,sorted_index] = sortrows({dir_struct.name}');
set(handles.lstFn,'String',sorted_names,...
        'Value',1)
set(handles.txtPath,'String',dir_path)


% --- Executes during object creation, after setting all properties.
function txtPath_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtPath (see GCBO)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


function txtGamma_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)

% Update lblDeg
degrees = radtodeg(str2double(get(handles.txtGamma, 'String')));
text_deg = sprintf('(%2.1f deg)',degrees);
set(handles.lblDeg, 'String',text_deg);

popPlotType_Callback(hObject, eventdata, handles);


function txtGamma_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtGamma (see GCBO)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end


% --- Executes on button press in chkUpdate.
function chkUpdate_Callback(hObject, eventdata, handles)
% hObject    handle to chkUpdate (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
lstFn_Callback(hObject, eventdata, handles);


function txtAlphaMin_Callback(hObject, eventdata, handles)
```

```
% handles    structure with handles and user data (see GUIDATA)
set(handles.chkUpdate, 'Value', get(handles.chkUpdate, 'Min'))
popPlotType_Callback(hObject, eventdata, handles);

% --- Executes during object creation, after setting all properties.
function txtAlphaMin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtAlphaMin (see GCBO)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function txtAlphaMax_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)
set(handles.chkUpdate, 'Value', get(handles.chkUpdate, 'Min'))
popPlotType_Callback(hObject, eventdata, handles);

function txtAlphaMax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtAlphaMax (see GCBO)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function txtKappaMin_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)
set(handles.chkUpdate, 'Value', get(handles.chkUpdate, 'Min'))
popPlotType_Callback(hObject, eventdata, handles);

function txtKappaMin_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtKappaMin (see GCBO)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function txtKappaMax_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)
set(handles.chkUpdate, 'Value', get(handles.chkUpdate, 'Min'))
popPlotType_Callback(hObject, eventdata, handles);

function txtKappaMax_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtKappaMax (see GCBO)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end

function txtFz_Callback(hObject, eventdata, handles)
% handles    structure with handles and user data (see GUIDATA)
set(handles.chkUpdate, 'Value', get(handles.chkUpdate, 'Min'))
popPlotType_Callback(hObject, eventdata, handles);

function txtFz_CreateFcn(hObject, eventdata, handles)
% hObject    handle to txtFz (see GCBO)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```